REMARKS

This is intended to replace the previous response filed May 27, 2004 and is intended as a full and complete response to the Office Action dated August 25, 2004, having a shortened statutory period for response set to expire on September 25, 2004. Please reconsider the claims pending in the application for reasons discussed below.

Applicants have amended the specification. Applicants submit that the amended paragraphs do not introduce any new matter into this application. Consistent with MPEP § 2163.06 applicants specifically point out the support for the amendments to the specification.

The amendments for the First Amended Paragraph are supported by material from page 2, of application which provides: "Templates or parameterized types conventionally tell a compiler how to generate code at compile time. Templates may be likened to macros, as templates may be used to implement data structures or algorithms."

The Amendment to the Second Amended Paragraph is not new matter. Section 2163.07 part II, of the MPEP provides that an amendment to correct an obvious error does not constitute new matter. The error of referring to a "STACK" as a "first in first out" data structure would be obvious to one skilled in the art, as would the appropriate correction; namely, referring to a "STACK" as a "last in first out" data structure.

Figure 3 illustrates a decision box 32 with the label "template?" with an arrow labeled "no" leading to decision box 34. This figure supports the amendments made to the Third Amended Paragraph, and the amendments are consistent with Figure 3.

The sentence subsequent to the amendment made to the Fourth Amended Paragraph provides support for the amendment as it specifically refers to the *generation* of template source code:

"Notably, it is possible that no new template source code files 56 need be generated after an initial pass at step 42 ...." (P. 9, last paragraph)

Further, box 42 of Figure 4 (which the amended sentence describes) illustrates a step to "generate/update source files in template repository."

Applicants have amended the drawings. Applicants assert that this change does not introduce new matter, and is supported by the description of Fig. 1 as a computer system that, in part, "comprises development system 50, described in more detail with reference to Fig. 5."

Claims 1-26 remain pending following entry of this response. Claims 1, 4, 5, 7, 9 – 12, 14, 16 – 18 and 26 have been amended for reasons other than patentability and thus, should be given a full range of equivalents. Applicants assert that the amendments do not introduce new matter.

CLAIM OBJECTIONS

The Examiner objects to the claims as not being in accordance with 37 C.F.R. 1.126 and has renumbered the claims accordingly. When the application is ready for allowance, the Examiner will renumber the claims consecutively in the order in which they appear.

CLAIM REJECTIONS

Claims 1-26 stand rejected under 35 U.S.C. § 102(b) as being anticipated by *Faiman* (US 5,836,014, hereinafter *Faiman*). Applicants respectfully traverse the rejection, and, therefore, respectfully request that the objection be withdrawn.

As described by Applicants, templates are a programmatic mechanism that implements parametric polymorphism. In other words, templates provide a construct that developers may use to implement classes, expressions and behavior independent of the data types being operated upon. When a template is used with a particular set of parameters, a compiler must instantiate a version of the template using the same parameters used in the source code module. In other words, a different template instantiation is required for each occurrence of a template that references a different set of parameters. Applicants claim

methods, systems, apparatus, and computer-readable mediums used to automatically instantiate templates present in source code modules by generating template source code corresponding to each template and then compiling the template source code to object code that may be linked with other object modules on a target computer system. No such teaching is made by *Faiman*.

Faiman is directed to a compiler framework that includes front-ends that process source code written in different programming languages into an intermediate representation, a compiler core used to optimize the intermediate representation and a back end that may emit object code for different target computer systems. (*See Faiman*, Abstract.). The compiler framework taught by *Faiman* teaches a system that performs number of object code optimizations. *Faimain's* title is illustrative: "Method Of Constructing A Constant-Folding Mechanism in A Multilanguage Optimizing Compiler." The constant folding routine taught by *Faiman* is described on col. 21, line 56 – col. 24, line 59. *Faiman* describes additional optimizations based on effects classes (*Faiman*, col. 12, line 57 – col. 18, line 39) and induction variables *Faiman* (col. 18 line 40 – col. 21 line 55.)

Faiman also teaches the use of templates. However, the templates of *Faiman* are *code generation* templates that may be used to optimize object code. *Faiman* describes code generation templates as consisting of four pieces of information:

> 1. A result value mode (see the examples given in the Appendix) which encodes the representation of a value computed by the template's generated code.
>
> 2. A pattern tree which describes the arrangement of ILG nodes that can be coded by this template. The interior nodes of the pattern tree are IL operators; the leaves of the pattern tree are either value mode sets or IL operators with no operands.
>
> 3. A sequence of Boolean tests. All of these must evaluate to true in order for the pattern to be applicable.
>
> 4. An integer that represents the "cost" of the code generated by this template.

(*Faiman* col. 24 line 67 – col. 25, line 15.) The template with the lowest "cost" is used to generate object code. The code generation templates disclosed in *Faiman* are included as part of the compiler framework, not part of the source code modules, and are used to compare different possible arrangements of object code to select the most optimal.

> In addition, a feature of one embodiment is a method for doing code generation using code templates in a multipass manner. The selection and application of code templates occurs at four different times during the compilation process: (1) The pattern select or PATSELECT phase does a pattern match in the CONTEXT pass to select the best code templates;

(*Faiman*, col.4, lines 40 – 47.)

Faiman also discloses using a linker process to link together multiple object modules to create an executable binary image for a particular target system or architecture. (*Faiman* col. 6 33-35, FIG. 1 #25). In contrast, Applicants claim a method for automatic instantiation of templates by creating object files that may be linked by the linker of the target system. (Claims 5-9, 10 – 15, 26, 21 – 25.)

Accordingly, the *code generation* templates included in the complier framework described by *Faiman* do not teach, show or suggest the generation of template source code used for automatic instantiation of templates that is claimed by Applicant.

A discussion of specific claim recitations now follows:

Claim 1 recites a template instantiation portion configured to generate template source code for each occurrence of a template appearing in the set of source code modules. The Examiner suggests that this recitation is disclosed by *Faiman* at col. 5, lines 24-25 and line 42. Applicants respectfully submit that col. 5, lines 24-25 and line 42 are directed generally to the "retargetable optimizer" and "code generator" components of compiler framework described by *Faiman*. These components do not teach, show or suggest a template instantiation

portion that generates template source code claimed by Applicants. *Faiman, t*herefore, does not teach, show or suggest the recitations of Applicants' claim.

Claim 5 recites providing source code; extracting template information from the source code; [and] providing the template information to a template repository. The Examiner suggests that this recitation is disclosed by *Faiman* Fig. 1, #21, source code, #55 intermediate language graph, col. 28, lines 14-16. Applicants submit, however, that the intermediate language graph, col. 28, lines 14-16 depicted in Fig. 1, #55 is directed to the conversion of one intermediate representation of the source code being compiled into another form. More specifically, *Faiman* teaches multiple optimizations that may be made to the intermediate representation, not the generation of additional source code (i.e., the claimed template source code module).

Further, Applicants claim a template repository that stores template information that is constructed by storing the template information code in a template repository and using the stored information to generating template source code that is compiled into object code that may be provided to a target system for linking and execution. *Faiman*, therefore, fails to teach, show, or suggest the generation of additional source code, ultimately compiled into an object module that may be linked by a target system. Accordingly, *Faiman* fails to teach show or suggest the template repository claimed by Applicants.

Claim 5 also recites generating template source code in response to information from the template information. The Examiner suggests that this recitation is disclosed by *Faiman* Examples of template code at col. 77-128). Applicants submit that the "several examples here including very simple addition templates and very complicated addressing templates" (*Faiman*, col. 77-78) are representative of the *code generation* templates described by *Faiman*. Each code generation template may, or may not, be used by the compiler to optimize the object code generated by *Faiman*'s compiler framework. (*Faiman*, col.4, lines 40 – 47) In contrast, Applicants claim a method to automatically instantiate templates, by generating template source code from any programmatic template

occurrences appearing in the source code modules. Because the code generation templates disclosed by *Faimain* are directed to templates included in the compiler framework, *Faiman* fails to teach show or suggest generating additional source code (i.e., the template source code) dependent on what template occurrences appear within the source code modules.

Claim 10 recites generating at least one template information file from the source code modules using the cross compiler. The Examiner suggests this recitation is disclosed by *Faiman* Appendix, col. 69 and Fig 1. Applicants submit that *Faiman* Appendix, col. 69 is directed to the operations of an actions interpreter. *Faiman* describes that "there are three different action interpreters--the CONTEXT interpreter, the TNBIND interpreter and the CODE interpreter. The actions of each template are performed in three different passes of the compiler by the appropriate interpreter." (Faiman, col. 25 lines 61-63.) As above, the code generation templates are part of the compiler framework described by *Faiman*, not programmatic elements of the source code modules from which the claimed template information files are generated. *Faiman*, therefore, fails to teach, show or suggest generating the template information files from the source code modules claimed by Applicants.

Claim 10 also recites providing the at least one template information file to the template repository; [and] generating at least one template source code module from the at least one template information file. The Examiner suggests this recitation is disclosed by *Faiman*, col. 16, lines 35-37, which set out in full provides:

> **probably be inefficient. A more sophisticated implementation might be built around a hash table, so that multiple small stacks (possibly each concerned with only one or a few)**

Applicant respectfully submits that this citation has nothing at all to do with generating additional source code based on the occurrence of programmatic template elements appearing in a set of source code modules used to generate

template information files stored in a template repository that is claimed by the Applicants, and that the objection should, therefore, be withdrawn.

Claim 10 also recites wherein the object code is linkable on a target computer system. The Examiner suggests that this recitation is disclosed by *Faiman*, Fig. 1 #25 and col. 6, lines 33-36). Applicants submit that *Faiman*, Fig. 1 #25 and col. 6, lines 33-36 teaches away from the object code modules claimed by Applicants that are linkable *by* the target computer system. Fig. 1, illustrates the linker as not being part of the target system 25, and further, that the object code modules 23 are linked before being provided to target system 25. (*See Faiman*, col. 6, line 35 "... to form an executable to run on the target machine 25").

Claims 16 and 21 both claim generating template object code modules that are generated from template source code that in turn is generated from occurrences of template elements in the source code modules. After the cross compiler generates template object code, the template object code may be linked by linker on a target computer. The Examiner cites the same portions of *Faiman* that the Examiner suggests teach methods of using code generation templates to optimize code. The Applicants characterization of these portions are found above. Accordingly, *Faiman* fails to teach show or suggest automatic template instantiation as claimed by Applicants for the same reasons set forth above.

The secondary references made of record are noted. However, it is believed that the secondary references are no more pertinent to the Applicants' disclosure than the primary references cited in the office action. Therefore, Applicants believe that a detailed discussion of the secondary references is not necessary for a full and complete response to this office action.

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted,

Gero G. McClellan
Registration No. 44,227
MOSER, PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicant(s)

IN THE DRAWINGS:

The attached sheet of drawings includes changes to Fig. 1. This Replacement Sheet, which includes Figs. 1 and 2, replaces the original sheet including Figs. 1 and 2. In the Replacement Sheet showing Fig. 1, replace the label "device system" with the label "development system" for the block marked by reference number 50.